VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Detekce obsazenosti parkovacích míst pomocí algoritmu strojového učení bez učitele

Parking Lot Occupancy Detection by Unsupervised Machine Learning Algorithm

2015 Václav Bilský

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Václav Bilský**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Detekce obsazenosti parkovacích míst pomocí algoritmu strojového učení bez učitele
Parking Lot Occupancy Detection by Unsupervised Machine Learning Algorithm

Zásady pro vypracování:

V dnešní době je využíváno technik zpracování obrazu v mnoha oblastech dopravy. Jedním z problémů je i detekce obsazenosti parkovacích ploch. Získané informace je možno využít např. pro automatizované parkovací systémy nebo v inteligentních dopravních systémech. Cílem práce je vytvořit aplikaci pro detekci volných parkovacích míst. Detekci proveďte pomocí vybraného algoritmu strojového učení bez učitele. Ve své práci proveďte:

1. Popište zadaný problém.
2. Představte princip strojového učení bez učitele.
3. Analyzujte řešení a popište potřebnou teorii vybraného algoritmu.
4. Implementujte aplikaci s použitím knihovny OpenCV.
5. Otestuje výslednou aplikaci a zhodnoťte dosažené výsledky.

Seznam doporučené odborné literatury:

[1] Duda, Hart, Stork: Pattern Classification, 2000, ISBN: 978-0471056690
[2] Ghahramani: Unsupervised Learning, 2004

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Michael Holuša**

Datum zadání: 01.09.2013
Datum odevzdání: 07.05.2015

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA * Fakulta elektrotechniky a informatiky * OSTRAVA *

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne 7. Května 2015

.....Bilský.....

Tímto bych rád poděkoval svému svému vedoucímu práce, panu Ing. Michalovi Holušovi, za všechen věnovaný čas v konzultací i mimo ně, cenné rady a nové zkušenosti.

Abstrakt

Tato bakalářská práce je zaměřena na detekci obsazenosti parkovacích míst ze statických snímků. Práce obsahuje popis úprav snímků potřebných pro předzpracování obrazu, popis algoritmu strojového učení bez učitele, princip fungování klasifikátorů K-means a Expectation Maximization. Implementace těchto částí byla za pomocí OpenCV prakticky odzkoušena na reálných snímcích parkoviště a nakonec popsány výsledky.

Klíčová slova: OpenCV, C++, Canny, K-means, Expectation Maximization, úprava obrazu, detekce obsazenosti, strojového učení bez učitele

Abstract

This bachelor thesis is focused on detecting the occupancy of parking spaces from still images. Contains a description of editing images needed for image pre-processing, description of unsupervised machine learning algorithm, the operating principle of classifiers K-means and Expectation Maximization. Implementation of these segments were using OpenCV practically tested on real images park and finally describes the results.

Keywords: OpenCV, C++, Canny, K-means, Expectation Maximization, picture adjustment, detecting the occupancy, unsupervised machine learning

Seznam použitých zkratek a symbolů

OpenCV	–	Open Source Computer Vision
OSS	–	Open-Source Software
BSD	–	Berkeley Software Distribution
GPU	–	Graphic Processing Unit
APU	–	Accelerated Processing Unit
CUDA	–	Compute Unified Device Architecture
OpenCL	–	Open Computing Language
NP	–	Nondeterministic Polynomial
EM	–	Expectation-Maximization
PSO	–	Particle Swarm Optimalization

Obsah

Seznam tabulek

Seznam obrázků

1 Úvod

Vidění, stejné slovo jak pro nejdůležitější a nejvyužívanější z naších smyslů, tak název pro vědecký obor zabývající se viděním a vnímáním pro stroje. Rozpoznávání věcí, pro nás naprostá samozřejmost díky naší inteligenci a schopnosti učení, pro počítače je takovéto univerzální řešení těžko řešitelný problém, a proto je zaměřeno jen na konkrétní, nebo podobné účely. Abychom získali nějaká data z obrazu, dokonce i tam, kde by bylo lidské oko neefektivní, nebo dokonce nepoužitelné, musíme obraz zpracovávat algoritmy, které nám dodají už určitá data, se kterými můžeme dál pracovat.

Obor počítačové vidění vznikl k jedinému účelu. Vznikl pro to, aby nám ulehčil práci poskytováním informací pro lepší efektivitu, šetří nám čas i peníze. Nyní jsou díky tomu technologie, které za nedlouho budou i zcela běžné a všem přístupné, jako samořídicí auta, nebo rozpoznávání věcí a kontextu z libovolného obrázku. Jako jsou ty současné, která ještě před pár lety nebyla, jako je automatické parkování, rozpoznávání vad v průmyslové výrobě, čtecí zařízení, ovládání her a mnoho dalších. Jednou z takových věcí je právě i detekce parkovacích míst.

Tato práce je zaměřena právě na detekci obsazenosti parkovacích míst pomocí strojového učení bez učitele. Rozebrání možností detekce a popsání jednotlivých algoritmů. Odzkoušení kódu na reálných fotografiích reálného parkoviště za různých podmínek a prezentace výsledků.

V první části s názvem Problematika a metody detekce jsou popsány dnešní technické možnosti detekce míst a proč je vhodné je použít detekci pomocí kamer. V další části je popsána open-source knihovna OpenCV, která se zabývá právě zpracováním obrazu spolu s nezbytným předzpracování obrazu pro následovné použití algoritmů, hlavně odstranění zkreslení čočky, perspektivní transformace. Proto aby bylo docíleno rovnoměrného rozložení parkovacích míst, separování jednotlivých míst a pár dalších nezbytných úprav k zajištění co nejlepších výsledků. Poté jak na zpracování jednotlivých parkovacích míst, které je zajištěno pomocí detekce hran a následného vyhodnocení dat na konkrétní výsledky v kapitole Detekce objektů. Nakonec přiblížení pojmu strojového učení ve stejnojmenné kapitole. A jako poslední, prezentace výsledků z testování ze statické kamery nad školním parkovištěm a závěr.

2 Problematika a metody detekce

Detekovat volná parkovací místa je někdy zcela nepotřebné z důvodů málo frekventovaného parkoviště třeba v menších městech, nebo třeba tam, kde je situace přehledná už při příjezdu. Ne vždy jsou ale takto ideální a přehledné situace a už méně časté v hustě zalidněných oblastech, kde je málo parkovacích míst, zástavba a hodně aut. Vznikají problémy, lidé musí hledat místo, ztrácí se čas, průjezdnost a celkově prostředky, které jsou v konečném součtu nemalé.

Je mnoho způsobů, jak detekovat a oznamovat prázdná místa. Dnes jsou hlavně nová podzemní parkovací místa vybavena informativními tabulemi a nad místy světlo upozorňující na jeho neobsazenost. Informace mohou být poskytnuty dále třeba poskytovatelům navigací či dopravních dat.

Kdysi byl systém odkázán jen na lidský faktor, což je neúnosné a chybové a u neplacených parkovišť i nesmyslné. Dnes je známo a v praxi už využíváno více systémů na detekci.

2.1 Detekce pomocí čidel

Detekovat se dá už dnes několika způsoby, nebo jejich kombinací.

- Nejrozšířenější a nejpoužívanější je dnes dobře známý závorový systém. Člověk přijede, vezme lístek, vjede. Problém je, že člověk neví, kde je volné místo a navíc je nepoužitelný u bezplatných parkovišť.
- Dalším způsobem je detekce magnetického pole, kdy je čidlo umístěno v asfaltu vozovky, tedy prakticky nepoužitelný systém na již postavená parkoviště. Čidla se musí navíc jednotlivě nakonfigurovat.
- Pak je tu detekce pomocí ultrazvukového měření vzdálenosti. Princip je založený na používání ultrazvukových senzorů. Ultrazvukové senzory jsou instalovány nad parkovací místa. Jedno čidlo na každé parkovací místo. Detekována vzdálenost od senzoru k překážce je následně vyhodnocena. Tento postup je velmi přesný, ale není nákladově efektivní, jak při instalaci a údržbě [2].

Metody jsou většinou používány spolu se systémem závor, mohou být doplněny o infračervené snímače detekující změny v infračerveném záření těles, které objekty vydávají a zajišťovat tak i dopravní informace o průjezdnosti. Tyto metody mohou být celkem

nákladné, čidel musí být velký počet, jednotlivě se musí nainstalovat, případně nakonfigurovat. Čidla musí komunikovat s centrální částí buď to drátově nebo bezdrátově, třeba pomocí bluetooth. I když samotná čidla moc energie nespotřebovávají, v celku se s nimi do provozních nákladů počítat musí. Mohou být sice napájena solárními panely, ale ty jsou z větší části nepoužitelné, jak v pozemních parkovištích, tak díky počasí. Mohou se vyskytnout situace, kdy se na místě nevhodně zaparkuje auto přes více pruhů nebo motorka. Poté si senzory nemusí poradit.

2.2 Detekce za pomoci kamer

Zpracování obrazu z videokamer je další možností, která může být použita pro automatickou detekci parkováních prostor. Použitím kamer můžeme dosáhnout nejen určení zda je parkovací místo obsazené, nebo ne. Může vykonávat i systém bezpečnostní, tím se náklady na instalaci obou systémů nebo i pozdější údržbu podstatně zmenší.

Je více možností způsobů detekce. Procesy se liší. Jeden ze způsobů je, že se může naučit oříznut obrázky automobilů a vytvořil klasifikátor pomocí Fuzzy c-means based classifier (FCMC) a upravení parametrů pomocí Particle swarm optimalization (PSO). Tento způsob je přesný, ale jde o komplexní přístup. Tady je potřeba obrovská sada dat vozů ze všech možných pohledů a světelných podmínek k vybudování klasifikátoru, takže není flexibilní, aby se mohl dostat do běžného použití [2].

Další může být za využití grafových řezů ze stereo obrazů, k čemuž jsou potřeba dvě kamery blízko sebe, rozdílem v obraze určí objekty vystupující do prostoru.

3 Úprava obrazu

Úprava obrazu je nezbytná, protože každá kamera má své vlastnosti a je umístěna v jiných úhlech a výškách. V této práci byla kamera na budově školy.

Aby byla vůbec možná detekce, je potřeba obraz upravit tak, aby každé parkovací místo bylo stejné, respektive, vstupní data při porovnávání byla stejná a mohly se později správně zpracovat a vyhodnotit.

V případě této práce se muselo upravit rovnou několik vlastností. Nejprve vlastnost způsobená snímanou kamerou a to zkreslení barelové, které způsobuje "prohnutí"obrazu, takže objekty nemají svůj skutečný tvar. Tedy, aby bylo parkoviště na snímku hranaté a ne zaoblené. Poté se musela provést perspektivní transformace, která zařídí, aby protilehlé strany parkoviště byly rovnoběžné, stejně dlouhé a s ostatními stranami svíraly úhel devadesáti stupňů. Tím je dosaženo toho, že každé parkoviště má stejnou velikost a tedy můžeme separovat každé místo tak, abychom na ně mohli použít algoritmus a poté data vyhodnotit.

3.1 OpenCV

K tomu, aby se dalo jednodušeji programovat a efektivněji využívat počítačové vidění slouží knihovna OpenCV, která obsahuje funkce pro práci s obrazem a umožní tak efektivní práci s daty. Neslouží jen k úpravě obrazu, ale i celkově k zpracování dat a díky těmto jejím výhodám se v této práci hojně využívá.

Název OpenCV je odvozen od anglických slov Open SourceComputer Vision Library. Je šířen pod licencí BSD, a proto je zdarma pro jakékoli využití včetně komerčního. Byla napsána a optimalizována pro C / C ++, kterých bylo v této práci při implementaci kódu využito, ale podporuje i jiné programovací jazyky jako Python a Java. Je multiplatformní, podporuje operační systémy Windows, Linux, Mac, iOS a Android. OpenCV byl navržen se zaměřením na běh aplikací v reálném čase. Podpora více jádrových výpočtů přes GPU a APU s pomocí dalších knihoven OpenCL a CUDA. Použití se pohybuje od interaktivního umění, zpracování a detekce objektů, až přes moderní robotiku.

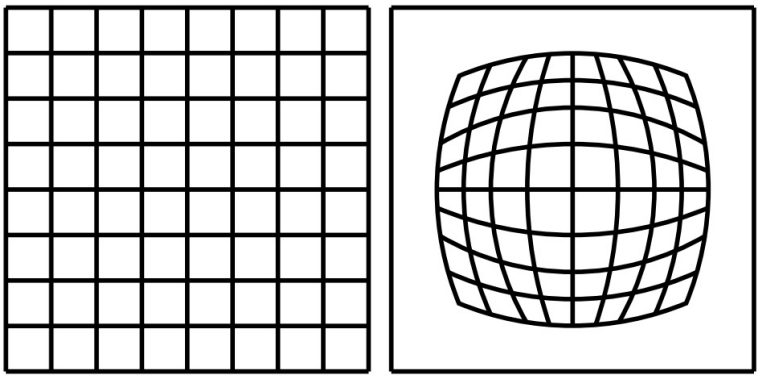
Právě díky takovýmto výhodám je světově nejrozšířenějším způsobem, jak zpracovávat obraz a proto se používá i v této práci.

3.2 Odstranění barelu

Vstupní obraz získaný kamerou nemusí být vždy takový jaký chceme. Na kvalitě vstupu záleží a nějaké úpravě se prakticky nevyhneme.

Díky optickým parametrů kamery mohou záběry nabývat geometrická zkreslení. Nejčastěji touto vlastností je takzvané radiální zkreslení fisheye, které vzniká u širokoúhlých objektivů s extrémně malou ohniskovou vzdáleností.

Jedním z typů tohoto je zkreslení je barel, jako je na obr. 1, které zapříčiňuje zvětšení středu obrazu, než jeho okrajových částí. Při zpracování obrázku jako v našem případě, by při detekci parkovacích míst a neodstranění barelu prakticky nebylo možné. Nedocházelo by k rovnoměrnému rozdělení jednotlivých míst a data by byla zkreslená.



Obrázek 1: Původní nezkreslený obrázek a jeho zkreslení typem barel [1]

Radiální symetrické zakřivení může být aproximováno s pomocí Taylorova polynomu:

$$\phi(r^2) = 1 + K1r^2 + K2^2 + ..., \tag{1}$$

kde $(r^2) = \overline{x}^2 + \overline{y}^2$ a $K1$ a $K2$ jsou koeficienty radiálního zkreslení. Souřadnice $\overline{x}$ a $\overline{y}$ jsou bezrozměrné, aby byly koeficienty nezávislé na velikosti vstupního obrazu.

Výsledná transformace se lze zapsat takto:

$$\begin{pmatrix} x_d \\ y_d \end{pmatrix} = \begin{pmatrix} x_n - c_u \\ y_n - c_v \end{pmatrix} \phi^{-1}(r^2) + \begin{pmatrix} c_u \\ c_v \end{pmatrix} \tag{2}$$

kde x_d a $y_d$jsou výsledné souřadnice, x_n a y_n původní souřadnice deformovaného obrazu a $c_u = w/2$ a $c_v = h/2$ jsou středy obrazu, kde w je šířka vstupního obrazu a h je jeho šířka.

Postupně procházíme jednotlivé pixely výstupního obrazu a vypočítáváme souřadnice korespondujících pixelů původního deformovaného obrazu [1].

Obrázek 2: Původní obrázek.

Obrázek 3: Obrázek bez zakřivení způsobeného čočkou

3.3 Perspektivní transformace

Jak už bylo naznačeno výše, nyní je obraz potřeba upravit tak, abychom získali ideálně rovnoměrné rozložení parkovacích míst. Problém je v tom, že vzdálené objekty jsou ve výsledném obrazu menší, než objekty stejné velikosti v popředí a tudíž z nich nemůžeme dostat stejně hodnotná data.

Na tuto úpravu se použije knihovna OpenCV, která obsahuje funkce, aby se mohla provést perspektivní transformace. V této části se provádí nezbytné geometrické transformace. Nemění obsah obrazu, ale deformují pixelovou mřížku a mapují tyto deformované sítě do cílového obrazu. Aby se zabránilo vzorkování artefakty, mapování se provádí v opačném pořadí, od místa určení ke zdroji. To znamená, že pro každý pixel o souřadnicích (x, y) cílového obrazu, funkce vypočítá souřadnice odpovídající "darovanému"pixelu ze zdrojového obrazu a zkopíruje z něho hodnotu pixelu a ty samé souřadnice x a y [3].

Funkce *getPerspectiveTransform(InputArray src, InputArray dst)*, kde: src - jsou souřadnice vrcholů čtyřúhelníku zdrojového obrazu, dst - jsou souřadnice odpovídající vrcholům čtyřúhelníku v cílovém obraze. Funkce počítá 3 x 3 matice perspektivní transformace takto:

$$\begin{bmatrix} t_i x'_i \\ t_i y'_i \\ t_i \end{bmatrix} = map\_matrix \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \tag{3}$$

kde $dst(i) = (x'_i, y'_i)$, $src(i) = (x_i, y_i)$,$i = 0, 1, 2, 3$.

Poté se aplikuje perspektivní transformace do obrazu metodou *warpPerspective(InputArray src, OutputArray dst, InputArray M, ...*, kde src - je vstupní obraz, dst – je výstupní obraz a M – je transformační matice velikosti 3x3. Funkce warpPerspective transformuje zdrojový obraz s použitím specifické matice [3]:

$$dst(x, y) = src\left(\frac{M_{11}x + M_{12}y + M_{13}}{M_{31}x + M_{32}y + M_{33}}, \frac{M_{21}x + M_{22}y + M_{23}}{M_{11}x + M_{32}y + M_{33}}\right) \tag{4}$$

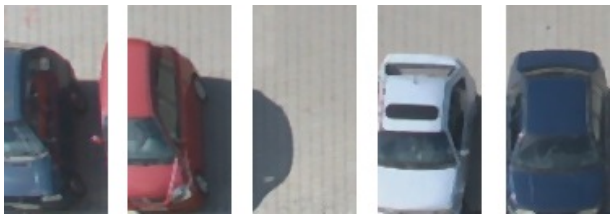
3.4 Nařezání jednotlivých parkovacích míst

Poté co se provede perspektivní transformace, a každé místo zabírá stejnou plochu, se můžou jednotlivá místa "nařezat". K tomuto opět poslouží knihovna OpenCV s funkcí *cvSetImageROI(image, cvRect(x, y, šířka, výška))*, kde parametry x a y jsou počáteční sou-

Obrázek 4: Obrázek po perspektivní transformaci

řadnice v obrazu odkud se bude "řezat"a hodnoty šířky a výšky jsou velikosti vyřezané části udávány v pixelech.

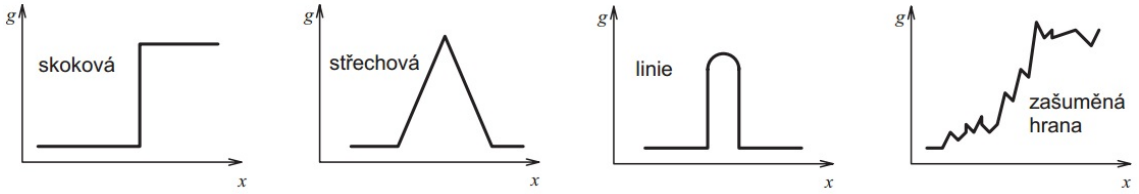
Souřadnice se musí zadávat ručně z důvodů, že se některá auta překrývají, zvláště u okrajů, kde zejména vyšší auta zasahují do vedlejšího stání.



Obrázek 5: Obrázek jednotlivých nařezaných parkovišť

4 Detekce objektů

Objekty se dají detekovat pomocí hran. Hrany se často uplatňují při zpracování obrazu, tedy i u počítačového vidění, v tomto konkrétním případě k detekci objektů, aut. Je dána vlastnostmi obrazového elementu a jeho okolím, její funkcí je označit body/pixely, ve kterých se ostře mění jas. Tyto body jsou obvykle uspořádány do rovných, a nebo zakřivených úseček, tedy tam, kde je okraj objektu. Místa, kde jsou hrany, nesou více informací než jiná místa v obraze. Hranový bod je bod s velkým gradientem, některé body v obraze jsou tedy hranové a jiné ne. Nalezené body jsou dle různých kritérií prahovány do absolutních hodnot.



Obrázek 6: Jasové profily v okolí hranových bodů [9]

Jak je vidět na obrázku 6, kde parametr g vrací intenzity hranových bodů. Tedy u prvních tří grafů jsou ideální výsledky, kde se vyskytují hrany. Na reálném obrazu se setkáme spíše se čtvrtou variantou, kde je obraz je zašumělý. Šumu se dá zbavit například Gaussovým filtrem, nebo funkci medián. Na některé části obrazu, které jsou ostřejší se můžou použít jiné hodnoty než na ostatní části, protože ne všechny části obrazu mají stejný počet detailů a tyto detaily poté znehodnocují celý výsledek. V konkrétním případě je zámková dlažba, která je přímo pod kamerou viditelná, obzvláště když jsou spáry mokré, kdežto dál nikoli. V takovém případě detektor detekuje velký počet hran na prázdném místě[9].

Hrany musejí být spolehlivě detekovány. Neúplné hrany, špatně lokalizované hrany a nejednoznačně detekované hrany se negativně promítnou do výsledku.

Algoritmů na detekci existuje více, které jsou založeny na různých principech lišící se přístupem k obrazu a definicí hrany a tedy v rozdílnosti výsledků.

4.1 Histogram Orientovaných Gradientů - HOG

Histogram Orientovaných Gradientů (Histogram of Oriented Gradients - HOG) se používá v počítačovém vidění a zpracování obrazu za účelem detekce objektů. Je považován za jednu ze základních metod pro detekci objektů.

Základní myšlenkou je, že vzhled a tvar objektu může být často lépe charakterizován pomocí intenzity gradientů či směru hran, a to i bez přesné znalosti příslušných gradientů, nebo pozic hran. V praxi je provedeno rozdělením okna snímku na stejně malé prostorové oblasti ("buňky") a pro každou buňku,se vytvoří 1-D histogramu gradientů vytvořený z pixelů v buňce, buďto se směry gradientů, nebo orientací hran. Pro lepší stálost osvětlení, stínování atd., je vhodné před dalšímy výpočty použít kontrastní normalizaci. To může být provedeno tím, že shromáždí měření nejen z lokálního histogramu, ale i z okolních buněk. Tato větší oblasti se nazývá blok. Potom se pomoci této hodnoty bloku standardizují ostatní hodnoty buněk uvnitř bloku [18].

4.1.1 Výpočet gradientu

Prvním krokem pro výpočet v předzpracování obrazu pro detektory příznaků je zajištění normalizovaných hodnot barev a gama hodnot. Tento krok může být ve výpočtu HOG deskriptoru vynechán, následná deskriptorová normalizace v podstatě dosáhne stejného výsledku. Předzpracování obrazu tak poskytuje jen malý vliv na výkon. Místo toho je první krok výpočtu výpočet hodnot gradientu. Nejběžnější metodou je aplikace 1-D se středem bodu diskrétní derivované masky v jednom nebo v obou z horizontálních a vertikálních směrů. Konkrétně, tato metoda vyžaduje filtrování barvy nebo intenzity dat obrázku s následujícími filtrací kernelu:

$$[-1, 0, 1], [-1, 0, 1]^T \tag{5}$$

4.1.2 Orientace obrázkových bodů

V druhém kroku výpočtu je vytvoření histogramů buněk. Každý pixel v buňce má svou hodnotu a tou ovlivňuje základní orientaci histogramového kanálu na základě hodnot zjištěných ve výpočtu gradientu. Samotné buňky mohou být buď obdélníkového, nebo radiálního tvaru, a kanály histogramu jsou rovnoměrně rozloženy od 0 do 180 nebo 0 do 360 stupňů. V závislosti na tom, zda je uvedený gradient se znaménkem nebo bez něho.

V praxi se zjistilo, že gradienty bez znaménka používané ve spojení s devíti histogramovými kanály provádí nejlepší detekci. Pokud jde o velikosti hodnot, pixel může být buď samotná hodnota gradientu, nebo některá velikost funkce.

4.1.3 Bloky deskriptoru

Aby se vzaly v úvahu změny v osvětlení a kontrastu, intenzita gradientů musí být lokálně normalizována, což vyžaduje seskupení buněk dohromady do větších, prostorově propojených bloků. HOG deskriptor je pak zřetězený vektor složek normalizovaných buněčných histogramů ze všech bloku regionů. Tyto bloky se typicky navzájem překrývají, což znamená, že každá buňka přispěje více než jednou na vytvoření konečného deskriptoru. Existují dvě hlavní geometrie bloků: obdélníkové bloky R-HOG a kruhové bloky C-HOG. R-HOG bloky jsou obvykle čtvercové mřížky, reprezentované třemi parametry: počet buněk na blok, počet pixelů na jednu buňku, a počet kanálů na histogram buňky. C-HOG lze nalézt ve dvou variantách: na ty s jednou centrální buňkou a na ty, které mají úhlově rozdělenou centrální buňku. Kromě toho tyto bloky C-HOG mohou být popsány se čtyřmi parametry: počet úhlových a kruhových výřezů, poloměr středového výřezu, a faktor expanze pro poloměr dalších kruhových výřezů.

4.1.4 Normalizace bloku

Jsou čtyři různé metody pro normalizaci blokovou. Nechť v je ne-normalizovaný vektor obsahující všechny histogramy v daném bloku, $||v||_k$ jsou jeho k-norma pro $k = 1, 2$ a $e$je nějaká malá konstanta (její hodnota je zanedbatelná). Potom normalizační faktor může být jedním z následujících:

$$L2 - norm : f = \frac{v}{\sqrt{[||v||_2^2 e^2]}} \tag{6}$$

L2-hys: L2-norm a následné výřezy (omezující max. hodnoty 0,2 V) a renormalizace, stejně jako v: (7)

$$L1 - norm : f = \frac{v}{(||v||_1 + e)} \tag{8}$$

$$L1 - sqrt : f = \sqrt{[\frac{v}{||v||_1 + e}]} \tag{9}$$

Kromě toho schéma L2-hys může být vypočten, že se nejprve vezme L2-norm, ořeže výsledek, a potom renormalizuje. V praxi se zjístilo, že L2-hys, L2-norm, a L1-sqrt systémy poskytovaly podobný výkon, zatímco L1-norma poskytuje o něco méně spolehlivý výkon. Nicméně, všechny čtyři metody vykazovaly velmi významné zlepšení v porovnání s nenormalizovanými daty [6].

Obrázek 7: Ukázka hogu. [17]

Tady je znázornění jak funguje histogram orientovaných gradientů na obrázku 7 V prvním sloupci jsou auta, tak jak je vidí kamera, ve druhém je ukázka změny velikosti, ve třetím je vidět histogramy orientovaných gradientů, každý z nich připadá jedné buňce a ve čtvrtém sloupci jsou podrobněji gradienty znázorněni s jejich velikostmi a směrem.

4.2 Canny

Canny je detektor hran, byl vyvinut Johnem F. Canny v roce 1986. Algoritmus Canny se zaměřuje na splnění tří hlavních kritérií:

Nízká míra chyb, detekující pouze existující hrany.

Dobrá lokalizace, kdy vzdálenost mezi pixely hran detekovaných a reálných jsou minimalizovány.

Minimální odezva, zajišťující pouze jednu reakci detektoru na každou hranu.

Používá vícestupňový algoritmus pro detekci širokého rozsahu hran:

1. Před manipulací s algoritmem Canny, se nejprve musí použít Gaussianův vyhlazovací filtr, který obraz trochu rozmaže. Dojde tak k odstranění šumu obrazu. Cílem filtrace je eliminace šumu. Protože šum na snímku jsou vysokofrekvenční část obrazu, tak se šum může odstranit ve frekvenční oblasti. Canny operátor nejprve provede filtrování obrazu s dvojrozměrnou Gaussovou funkci, která je následující [15]:

$$G(x,y) = \frac{1}{2\pi\sigma} e^{-(\frac{(x^2+y^2)}{2\sigma})} \tag{10}$$

   kde x je vzdálenost od počátku v horizontální ose, y je vzdálenost od počátku ve svislé ose, a σ je standardní odchylka Gaussova rozdělení [11].

2. Nalezení gradientu intenzity obrazu. Za tímto účelem se používá, analogická procedura k Sobelu:

   (a) Použití dvojice konvulčních masek x a y:

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}, \; G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \tag{11}$$

   (b) Nalezení velikosti a směru gradientu s:

$$G = \sqrt{G_x^2 + G_y^2} \qquad \theta = \arctan(\frac{G_y}{G_x}) \tag{12}$$

   Směr je zaokrouhlena na jeden ze čtyř možných úhlů (konkrétně 0, 45, 90 nebo 135).

3. Aplikace Non-maximum potlačení. Tím se odstraní pixely, které nejsou považovány za součást hrany. Proto pouze tenké čáry (kandidátské hrany) zůstanou.

4. Poslední krok je hystereze, kdy Canny používají dvě prahové hodnoty (horní a dolní):

   (a) Pokud je gradient pixelu větší než horní prahová hodnota (treshold1), obrazový bod je akceptován jako hrana.

   (b) Je-li hodnota gradientu pixelu pod dolní prahovou hodnotu (treshold2), pak je odmítnut.

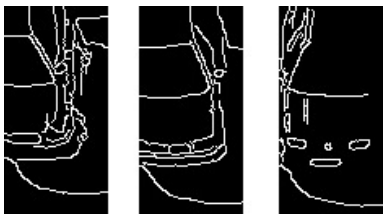
(c) V případě, že gradient pixelu je mezi dvěma prahovými hodnotami, pak bude akceptován pouze tehdy, pokud je spojen k pixelu, který je vyšší než horní prahová hodnota.

Implementace algoritmu Canny v kódu: *void Canny(InputArray image, OutputArray edges, double threshold1, double threshold2, int apertureSize=3, bool L2gradient=false))*

Kde jsou parametry:

- image – jednokanálový 8-bitové vstupní obraz.
- edges – výstupní mapa hran má stejnou velikost a typ, jako vstupní obrázek.
- threshold1 – první prahová hodnota hysterezní procedury.
- threshold2 – druhá prahová hodnota hysterezní procedury.
- apertureSize - velikost pro operátor Sobel().
- L2gradient – Příznak udávající $L_2\ norm = \sqrt{(dI/dx)^2 + (dI/dy)^2}$ by měl být použit na výpočet rozsahu gradientu obrazu (L2gradient=true), nebo zda je výchozí L_1 norm $=|dI/dx| + |dI/dy|$ je dostačující (L2gradient=false) [14].

Výstup Cannyho vypadá následovně.



Obrázek 8: Ukázka výstupu detektoru hran Canny

Detektor hran canny se použije až nyní z důvodu, že se může nastavit pro jednotlivá parkoviště zvlášť. Kdyby byl použit dříve nebyla by další možnost dodatečných úprav.

Parkovací místa v popředí jsou detailněji (z větší blízky) zachycena kamerou a jdou na nich vidět vzory zámkové dlažby, které Canny vyhodnotí jako hrany, takže prázdné místo má přibližně stejný počet hran jako zaplněné. Tím že se spouští detektor hran až nyní, tak je tu možnost tyto hrany lépe eliminovat. První způsob je tyto parkovací místa trochu rozostřit pomocí vyhlazovacího filtru Gaussian. Další

způsob je nastavení prahových hodnot na přísnější vyhledávání hran. Nebo, jako v konkrétním případě, obojím najednou, čímž se pozorováním a ověřením kontrol hodnot dosáhlo nejlepšího výsledku.

Nyní se může použít OpenCV funkce *countNonZero(ObrazekParkovacihoMista)*, která vrátí počet hran.

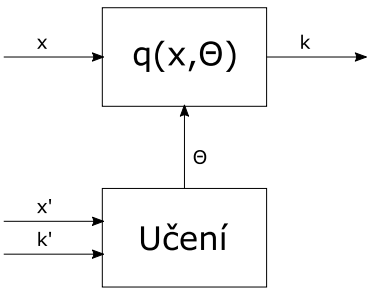
5 Strojové učení

Pokud jde o stroje, můžeme volně říct, že se stroj učí, kdykoli se změní jeho struktura, program nebo data, na základě svých vstupů nebo v reakci na nějakou externí informaci. A to takovým způsobem, že jeho očekávaný budoucí výkon se zlepší. Některé z těchto změn, jako je například přidání záznamu do databáze, spadají pod další obory a není nutně je nazývat učením. Ale, například, když se výkon stroje na rozpoznávání řeči zlepšuje po poslechu několika vzorků řeči od více osob, v tomto případě zcela oprávněně můžeme říci, že se stroj naučil [4].

Strojové učení se zabývá vývojem počítačových algoritmů a technik, které jsou schopné, učit se, tedy, automaticky se zlepšit na základě zkušeností. Každá metoda strojového učení se skládá ze dvou kroků, výběru kandidátského modelu, a poté, odhadováním parametrů modelu používající algoritmus učení a dostupných údajů. Velmi často jsou výběr modelu a odhad parametrů kombinovány v opakujících se procesech, a v mnoha případech, výběr modelu byl proveden pouze jednou intuitivně a empiricky. Jinými slovy, uživatel zvolí model empiricky, a poté, využívá algoritmus učení k odhadu parametrů modelu.

Pro algoritmy strojového učení existuje vícero způsobů jak data klasifikovat. Na vstupu jsou posílány data z takzvané trénovací množiny a sledují se výstupní hodnoty. Existují dva základní typy učení.

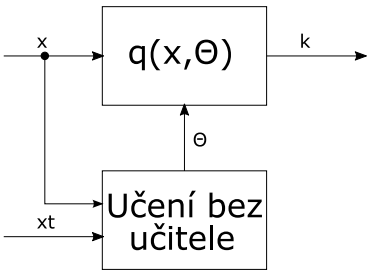
- Jedna velká kategorie je učení s učitelem, kde by měl model vytvářet přibližné mapování mezi vstupem a výstupem z daných dat, obvykle známý jako regrese nebo zařazení. Při učení s učitelem existuje nějaké vnější kritérium, klasifikátor určí zda je výstup správný. Schéma vypadá následovně:



Obrázek 9: Učení s učitelem

N obrázku 10 je znázorněné schéma jak takové učení bez učitele probíhá, kde x' jsou známá trénovací data a k' je známý výsledek, x jsou vstupní data, která chceme vyhodnotit přes klasifikátor porovnávající data od učitele $\theta = učení(x, k)$. Nakonec k jsou výstupní data z klasifikátoru.

- Učení bez učitele patří do druhé kategorie učících algoritmů. Shlukování (klasifikace) dat je typický způsob učení bez učitele, kde jsou dané sady dat rozděleny do různých podskupin (clusterů), tak, že data v každá podmnožina sdílejí některé společné rysy (podobnost) definované měřením vzdáleností. Při učení bez učitele žádné vnější kritéria neexistují a celé učení je založeno na informacích získaných během vlastního procesu učení.



Obrázek 10: Učení bez učitele

kde xt jsou další potřebná vstupní data na správné clusterování (viz. testování), k jsou výstupní data z klasifikátoru, který porovnává vstupní data s daty od učení bez učitele θ=*učení*(x, xt) [5] [6].

5.1 Učení bez učitele

Tato práce se věnuje právě učení bez učitele. Učení bez učitele se snaží najít neznámé struktury v neoznačených datech. Jeho použití je vhodné zrovna tam, kde jsou spousty trénovacích dat, nebo kde je neznámá klasifikace dat. Také je tu možnost komprimace dat díky tomu, že se nahradí rozsáhlé datové soubory několika málo významnými reprezentanty [7].

U učení bez učitele nemusí být předem jisté, zda patří do nějaké skupiny známých shluků. Úkolem je klasifikovat všechny objekty zahrnuté do analýzy. Tento postup se označuje jako shluková analýza. Učení je možné definovat pro stroje pomocí podmínek.

Podmínek může být více, mohou být složité, ale taky ulehčí analýzu.

5.2 Shluková analýza

Shluková analýza patří mezi metody učení bez učitele. Používá ke klasifikaci objektů a cílem je v dané množině objektů nalézt její podmnožiny, nebo-li shluky objektů, přičemž objekty mohou obsahovat více různých nesouvisejících dat. Snaží se nalézt objekty v shluku, které by si byly navzájem podobnější a přitom dostatečně jiné, než objekty mimo tento shluk. Shlukové metody se můžou rozdělit na hierarchické a nehierarchické, podle cílů, ke kterým směřují.

Hierarchické shlukování je systém průniku každých dvou podmnožin - shluků jedné množiny, které nejsou navzájem stejné a nejsou prázdné. Průnikem těchto podmnožin je buď to prázdná množina a zároveň v množině existuje alespoň jedna dvojice podmnožin, jejichž průnikem je jedna z nich. Nastane-li tento případ, pak se jedná o systém hierarchický. Hierarchického shlukování lze rozlišit na přístup divizní a aglomerativní. U divizního se vychází z jednoho shluku a ten se dále dělí. Aglomerativní vychází z více shluků o jednom členu a ty se spojují.

Nehierarchické shlukování je systém opět navzájem různých, neprázdných podmnožin jako u hierarchického systému, v němž ale průnikem každých dvou podmnožin není žádná z nich, tedy kde jsou shluky množiny disjunktní [13].

5.2.1 K-means

Pokud jsou již k dispozici data z nějakého detektoru hran, nyní je potřeba je nějak vyhodnotit. Může se použít třeba K-means algoritmus.

K-means nebo je též používán název "Lloydův algoritmus"a nebo též "MacQueenova metoda"podle MacQueena, který tento algoritmus publikoval v roce 1967. Algoritmmus K-means je nehierarchistická metoda[10].

K-means používá Euklidovskou vzdálenost a $\mu_j$je aritmetický průměr bodu ve shluku (střední hodnota ve třídě) – etalon.

Algoritmus K-means má předem daný počtu shluků $K : C_1, C_2, C_3, ..., C_k$. Opakovaně hledá hodnoty vektorů tak, že minimalizuje střední odchylku mezi vstupními daty a vektory (vzdálenost bodu od etalonu shluku), které mají k těmto datům nejmenší euklidovskou vzdálenost a podle toho je postupně přiřazuje do jednoho ze shluků.

Vstupem je množina dat $x_1, x_2, ..., x_l$ a číslo K udávající počet vektorů μ_j, j = 1, ..., k. Nejprve se nastaví vektory $\mu_j, j = 1, ..., k$ podle vzorových bodů, které lze vybrat náhodně ze vstupní množiny objektů, nebo použitím nějaké vhodně zvolené heuristiky. Po nastavení počátečních hodnot se začnou opakovat následující dva kroky:

1. Klasifikace: Všechna data $x_i, i = 1, ..., l$, se klasifikují do shluků podle vektorů $\mu_i, i = 1, ..., k$, na základě euklidovské vzdálenosti. Tedy data x_i jsou přiřazeny do shluku y_i podle $y_i = argmin_j||x_i - \mu_j||$.

2. Přepočet vektorů μ_j: Vypočtou se nové hodnoty vektorů μ_j , které se vypočítají pomocí vztahu

$$\mu_j = \frac{1}{l_j} \sum_{i=1, y_i=j}^{l} (x_i), \tag{13}$$

kde l_j je počet vzorů x_i klasifikovaných v druhém kroku do třídy určené vektorem μ_j.

Přiřazování a přepočítání se provede pokaždé, dokud se nevyberou všechny objekty. Klasifikace a přepočítávání vektorů se opakuje do té doby, dokud se alespoň jeden vektor x_i neklasifikuje do jiné třídy než byl klasifikován v předešlém kroku. Obecně platí, že s konečným časem K-means konverguje k pevnému bodu. Po přiřazení všech bodů jsou počáteční vzorové body výsledné těžiště shluků [13].

Výhodou je rychlost a jednoduchost, dá se použít na velké množství dat. Prvky se mohou postupně přeskupovat mezi shluky. V konečném počtu kroků konverguje k nějakému řešení, těch ovšem může být více.

Nevýhodou je, že výsledky jsou ovlivněny počáteční podmínkou, výběrem vzorových bodů nebo zvolenou heuristikou, a protože po přiřazení každého bodu dochází k okamžitému přepočítání těžiště, je výsledek také ovlivněn původním pořadím objektů. Pokud se ve výsledku očekávají překrývající se shluky, pak K-means není vhodný, protože jeden objekt je přiřazen právě jednomu shluku.

Přítomnost izolovaných objektů ležících mimo ostatní má velký negativní vliv na výsledek, o tomto problému se bude mluvit v závěru, konkrétně, proč nemůžeme rovnou vyhodnocovat jen jeden snímek parkovacího místa [8].

Následující funkce K-means je minimalizována:

$$f = \sum_{j=1}^{K} \sum_{x_j}^{K} ||x - c_j||^2 \tag{14}$$

kde ||.|| je zvolena vzdálenost měření mezi datovým bodem x a centrem (c_j) shluku C_j, K je počet shluků [5].

Implementace K-means v kódu: *double kmeans(InputArray data, int K, InputOutputArray bestLabels, TermCriteria criteria, int attempts, int flags, OutputArray centers=noArray())*

Kde jsou parametry:

- data - Data na shlukování, jeden řádek na každý vzorek.
- K - Číslo udávající počet shluků.
- bestLabels - Výstup, ukládá číslo shluku do pole pro každý vzorek.
- criteria - Kritéria pro ukončení algoritmu, pokud bylo dosaženo maximálního počtu opakování nebo požadované přesnosti, kde přesnost je specifikována jako criteria.epsilon. Pak jakmile se každé těžiště shluků pohybuje pod hodnotou criteria.epsilon na některém opakování, algoritmus se zastaví.
- attempts - Určuje počet opakování algoritmu v různých počátcích, algoritmus vrací labely, které vykazují nejlepší hodnoty.
- flags - Jsou metody na vybírání náhodných počátečních shluků.
- centers - Výstupní matice s daty o těžištích shluků, řádek na každé těžiště. [12]

5.2.2 Expectation Maximization

Zkráceně metoda EM z anglického názvu Expectation Maximization, je algoritmus odhadující parametry multirozměrné pravděpodobnosti hustoty funkce ve formě Gaussianově roztřídění směsí se specifickým číslem směsí.

Předpokládejme sadu N příznakových vektorů $x_1, x_2, ..., x_N$ z d-rozměrný euklidovského prostoru vypracované ze Gaussova směsi:

$$p(x; a_k, S_k, \pi_k) = \sum_{k=1}^{m} \pi_k p_k(x), \quad \pi_k \geq 0, \quad \sum_{k=1}^{m} \pi_k = 1,$$

$$p_k(x) = \varphi(x; a_k, S_k) = \frac{1}{(2\pi)^{d/2} \mid S_k \mid^{1/2}} exp\left\{-\frac{1}{2}(x-a_k)^T S_k^{-1}(x-a_k)\right\}, \quad (15)$$

kde m je počet směsí, p_k je normální hustota rozdělení se středem a_k a kovarianční matice S_k, π_k je váha k-té směsi. Vzhledem k počtu směsí M a vzorky $x_i, i = 1..N$ algoritmus najde odhad maximální pravděpodobnosti-(MLE) všech parametrů směsi, to znamená, že a_k, S_k a π_k:

$$L(x, \theta) = logp(x, \theta) = \sum_{i=1}^{N} log\left(\sum_{k=1}^{m} \pi_k p_k(x)\right) \to \max_{\theta \in \Theta},$$

$$\Theta = \left\{(a_k, S_k, \pi_k) : a_k \in R^d, S_k = S_k^T > 0, S_k \in R^{d \times d}, \pi_k \geq 0, \sum_{k=1}^{m} \pi_k = 1\right\}. \quad (16)$$

EM algoritmus je iterační procedura. Každá iterace zahrnuje dva kroky. V prvním kroku (krok Exception nebo E-step), můžete se najít pravděpodobnosti $P_{I,K}$ (označována jako $\alpha_{I,K}$ v níže uvedeném vzorci) vzorku i patřit do směsi k používající aktuálně dostupnou odhad parametrů směsi:

$$\alpha_{ki} = \frac{\pi_k \varphi(x; a_k, S_k)}{\sum_{j=1}^{m} \pi_j \varphi(x; a_j, S_j)}. \quad (17)$$

. V druhém kroku (krok Maximalizace nebo M-step), odhady parametrů směs se zpřesňují na základě vypočtené pravděpodobnosti:

$$\pi_k = \frac{1}{N}\sum_{i=1}^{N} \alpha_{ki}, \quad a_k = \frac{\sum_{i=1}^{N} \alpha_{ki} x_i}{\sum_{i=1}^{N} \alpha_{ki}}, \quad S_k = \frac{\sum_{i=1}^{N} \alpha_{ki}(x_i - a_k)(x_i - a_k)^T}{\sum_{i=1}^{N} \alpha_{ki}} \quad (18)$$

Případně, algoritmus může začít od N-tého kroku, jestliže je možno poskytnout počáteční hodnoty pro $P_{I,K}$. Další alternativou, kdy $P_{i,k}$ je neznámá, je použití jednoduššího shlukovacího algoritmu k předshlukování vstupních vzorků, a tak získat počáteční $P_{I,K}$. Často (včetně strojového učení) se používá k tomuto účelu algoritmus K-means [**?**].

Implementace Expectation Maximization v kódu: $boolEM :: train(InputArraysamples, OutputArraylogLikelihoods = noArray(), OutputArraylabels = noArray(), OutputArrayprobs = noArray())$;

Kde jsou parametry:

- samples - Matice s daty o parkovacích místech, řádek na jedno parkovací místo, sloupec na jednu dimenzi
- logLikelihoods - volitelné výstupní matice, která obsahuje logaritmus hodnoty pravděpodobnosti pro každý vzorek
- Labels - Volitelný výstup "class label"pro každý vzorek labely, které vykazují nejlepší hodnoty.
- probs - volitelné výstupní matice, která obsahuje posterior pravděpodobností každé Gaussova složky směsi uvedené každý vzorek.

6 Testování a výsledky

Při testováni byl brán ohled na různě klimatické a světelné podmínky. Dále na různé zaplněná parkovací místa, což se na začátku testování výrazně promítalo do výsledku tím, že se zcela zaplněná, nebo skoro zaplněná místa špatně vyhodnocovala díky tomu, že se algoritmy snažily rozdělit úzkou skupině velmi podobných údajů. Obdobný problém nastal u prázdného, nebo jen s několika zaplněnými místy.

Testovalo se pomocí dvou algoritmů na určení jednotlivých míst a to algoritmus k-mean a algoritmus expectation maximization. Data do nich byla plněna z histogramu orientovaných gradientů s nastavením HOGDescriptoru na velikost okna pro každé parkvací místo 64x128, velikost bloku 16x16, velikost krokovacího bloku 8x8, velikost buňky 8x8 a nbit nastaven na 9. Další hodnoty byly nechány původní s nimi se dosahovalo nejlepších výsledků. Dále bylo provedeno naplnění algoritmů nenulových bodů z detektoru hran canny.

K vyhodnocení výsledků bylo využito vzorce F1 score, které slouží jako měřítko testovací přesnosti, který vypadá takto:

$$F1Score = \frac{2TP}{(P + P')} = \frac{2TP}{(2TP + FP + FN)}, \tag{19}$$

kde $TP(TruePositive)$ je počet správně vyhodnocených výsledků, $FP(FalsePositive)$ je počet prázdných míst uvedených, jako zaplněných a $FN(FalseNegative)$ je počet zaplněných míst uvedených, jako nezaplněných.

6.1 První test

Různě zaplněná místa mají u algoritmu k-mean podstatný vliv na výsledek. Přítomnost pár izolovaných objektů ležících mimo ostatní neumí k-mean dobře zpracovat, i když tyto hodnoty jsou dost jiné a jedinečné od ostatních, tak při výpočtů clusterů nemají hodnoty takovou váhu, aby se u nich vytvořil samostatný center a tak jsou objekty přiřazeny špatně.

To se vyřeší přidáním dalších hodnot, tedy dalších parkovacích míst. Tím se dosáhne toho, že i když se bude vyhodnocovat prázdné, nebo plné parkoviště tak budou k dispozici další data na správné vyhodnocení a zařazení do clusteru. Tato data se dají nazvat jako trénovací. U algoritmu k-means se takto může může vytvořit matice, ke které se poté přidají data testovaného parkoviště. Ukládání a načítání trénovací matice je z hlediska výkonu a času mnohem výhodnější.

Trénovací matice byla vytvořena ze čtyř parkovišť pomocí HoGu, každé o 56 parkovacích místech. Ze dvou parkovišť byly výsledky horší než ze čtyř, protože patrně bylo ještě málo hodnot na správné clusterování a paradoxně ze sedmi taky, protože vznikalo takzvané přetrénování, což je, že k nakalibrování dojde nejen s ohledem na obecné trendy v trénovací množině, ale i všech náhodných procesů a složek zachycených v těchto datech [19].

Trénovací parkoviště byla vybrána tak, aby celkově obsahovaly přibližně polovinu zaplněných a polovinu prázdných míst, rovněž bez velkých klimatických a světelných podmínek, protože třeba díky slunci by se na volných místech tvořily stíny a ty by se vyhodnocovaly jako hrany. Celkově bylo prázdných 126 a obsazených 98, na mírném nepoměru mezi prázdnými a obsazenými místy už zase tolik nezáleží, jedná o už dostatek dat, aby se mohly správně vytvořit clustery.

Dále se testovalo vybraných 16 obrázků celkem 896 parkovacích míst, kde 455 míst bylo zaplněno a 441 bylo prázdných.

Úspěšnost byla 80,85% z 455 zaplněných míst bylo 168 chybných a ze 441 prázdných bylo 120 detekováno špatně.

6.2 Druhý test

Další test proběhl tak, že algoritmus k-mean byl naplněn pouze nenulovými body detektoru hran canny.

Tato jednorozměrná matice dosáhla nejlepšího výsledku a to 96.6%, bylo detekováno 26 chybě zaplněných míst a 23 chybně detekovaných aut.

6.3 Třetí test

Třetí test byl velmi podobný druhému s tím, že matici vyhodnocoval algoritmus expectation maximization.

Dosáhl i velmi podobného výsledku a to 96.42%, kde bylo detekováno 58 chybě zaplněných míst avšak pouze 4 chybně detekovaných aut.

6.4 Čtvrtý test

U čtvrtého testu nemohly být použity původní natrénované hodnoty z HoGu, protože expectation maximization má limitována hodnoty na výpočet hodnot pouze z určitého počtu dimenzí.

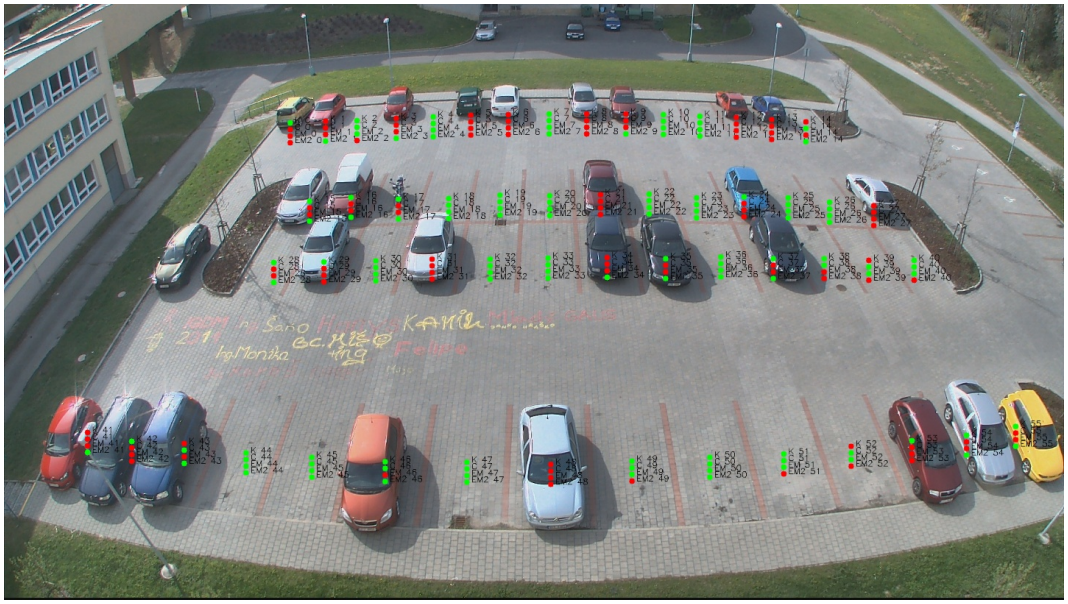
Muselo být použito nové nastavení HOGDescriptoru na velikost okna pro každé parkvací místo 64x128, velikost bloku 64x64, velikost krokovacího bloku 32x32, velikost buňky 32x32 a nbit nastaven na 9 což ve výsledku dalo 109 dimenzí. Bylo to nejdetailnější nastavení, které proběhlo úspěšně, avšak s nevalným úspěchem.

Úspěšnost byla 74.25% a špatně bylo detekováno 175 chybě zaplněných a 192 chybně detekovaných aut.

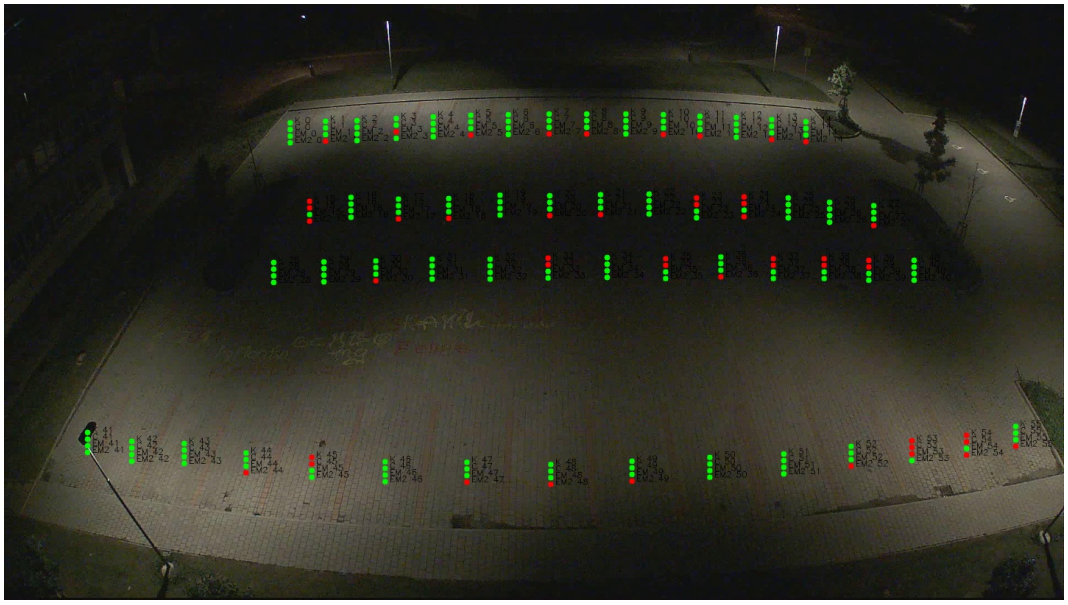
6.5 Porovnání testů

	Obsazené/Prázdné	FP/FN	Procenta úspěšnosti
Test1	455/441	120/168	80.85%
Test2	455/441	23/36	96.60%
Test3	455/441	58/4	96.42%
Test4	455/441	175/192	74.25%

Tabulka 1: Tabulka porovnávající testy.



Obrázek 11: Ukázka výsledků poloprázdného parkoviště.



Obrázek 12: Ukázka výsledků prázdného parkoviště ve tmě.

7 Závěr

Téma bakalářské práce se podařilo splnit. Práce teoreticky obeznamuje s detekcí objektů, princip strojového učení bez učitele, popis algoritmu histogram orientovaných gradientů, detektor hran canny expectation maximization. Byly odzkoušeny a vybrány nejlepší nalezené nastavení, které byly spuštěny a otestovány a následně popsány výsledky ve vytvořené aplikaci v jazyku c++ za pomoci knihovny OpenCV.

Testování ukázalo, že záleží na výběru zpracování dat a jaká a čím se budou zpracovávat, narazilo se na omezení expectation maximization v omezením počtu zpracovaných dimenzí, testovací matice se musela razantně upravit a data by se do provozu nedala použít. Testování proběhlo poměrně úspěšně kdy k vytvoření výsledku byl použit algoritmus k-mean, dosahoval úspěšnosti 80%. Nejefektivnější bylo, když se poslalo do algoritmů pouze jedna dimenze z detektoru hran canny, úspěšnost byla v obou algoritmech přes 96% avšak na takovýto druh informací by se stejným výsledkem šly použít lehčí cesty, ale důležité je, že se touto jednoduchostí ověřila teorie.

Řešení by bylo více, je mnoho nastavení a je velmi obtížné najít ty nejlepší a nejideálnější. Kdyby se měly porovnat výsledky učení bez učitele a s učitelem, učitel má mnohem lepší výsledky. Jde o to že se mu výsledků dostane a pak porovnává, kdežto učení bez učitele analyzuje a může zjistit třeba že parkovací místo něco blokuje. Výsledky by se patrně zlepšily, kdyby byla kamera vhodněji umístěna a nejednal o se venkovní parkoviště, protože klimatické podmínky hrají výraznou roli na zpracování výsledku.

Václav Bilský

8 Reference

[1] GAURA, Jan. *Odstranění geometrických zkreslení obrazu* [online].
Dostupné z: http://mrl.cs.vsb.cz/people/gaura/dzo/geom_distortion_cz.pdf

[2] KAIROEK, Choeychuen. *Available car parking space detection from webcam by using adaptive mixing features* [online].
Dostupné z: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6261917 //2012, strany: 12 - 16, ISBN: 978-1-4673-1920-1

[3] OpenCV. *Geometric Image Transformations* [online].
Dostupné z: http://docs.opencv.org/modules/imgproc/doc/geometric_transformations.html

[4] NILSSON, Nils *Instruction to machine learning*
prosinec 1998, University Stanford, CA 94305

[5] JIN, Yaochu *Pareto-Based Multiobjective Machine Learning: An Overview and Case Studies*
Květen 2008, strany: 397 - 415, ISSN: 1094-6977

[6] Petrus, Michal. *Rozšiřující SW mobotů, vycházející z poznatků etologie a genetiky (MODEL ADAPTACE CHOVÁNÍ)* [online]. 1997
Dostupné z: http://cyber.felk.cvut.cz/gerstner/eth/download/dpmp.pdf?PHPSESSID=56e6089a328ed3e96782089594178dba
Praha, ČVUT-FEL 1997

[7] HLAVÁČ, Václav. *Učení bez učitele* [online].
Dostupné z: http://cmp.felk.cvut.cz/~hlavac/Public/TeachingLectures/UceniBezUcitele.pdf
Praha, ČVUT

[8] HLAVÁČ, Václav. *Učení bez učitele* [online].
Dostupné z: http://cmp.felk.cvut.cz/~hlavac/Public/TeachingLectures/UceniBezUcitele.pdf
Praha, ČVUT

[9] HLAVÁČ, Václav. *Detekce hran* [online].
Dostupné z: http://cmp.felk.cvut.cz/~hlavac/TeachPresCz/11DigZprObr/22EdgeDetectionCz.pdf

Praha, ČVUT

[10] FLACH, Peter*MACHINE LEARNING The Art and Science of Algorithms that Make Sense of Data*//Listopad 2012, ISBN: 9781107422223

[11] OpenCV. *Gaussian blur* [online].
Dostupné z: http://en.wikipedia.org/wiki/Gaussian_blur

[12] OpenCV. *Clustering* [online].
Dostupné z: http://docs.opencv.org/2.4.9/modules/core/doc/clustering.html

[13] KELBEL, Jan. *Shluková analýza* [online].
Dostupné z: http://www.fd.cvut.cz/personal/nagyivan/Projekty/Classification/\ShlukovaAnalyza.pdf

[14] OpenCV. *Canny Edge Detector* [online].
Dostupné z: http://docs.opencv.org/doc/tutorials/imgproc/imgtrans/\canny_detector/canny_detector.html?highlight=canny

[15] HAO, Geng. *Improved Self-adaptive edge detection method based on Canny* [online].
Dostupné z: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6642801

[16] OpenCV. *Expectation Maximization* [online].
Dostupné z: http://docs.opencv.org/modules/ml/doc/expectation_maximization.html

[17] Wahyono, Van-Dung Hoang, Laksono Kurnianggoro, and Kang-Hyun Jo. *Scalable Histogram of Oriented Gradients for Multi-size Car Detection* [online].
Dostupné z: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7018581

[18] DALAL, Navneet. *Histograms of Oriented Gradients for Human Detection* [online].
Dostupné z: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1467360

[19] PŘIBYL, Ondřej . *Základní analytické metody* [online].
Dostupné z: http://zolotarev.fd.cvut.cz/ma/ctrl.php?act=show,file,22572

A Příloha

Součástí priloženého CD je:

- Elektronická verze této práce ve formátu PDF.
- Zdrojové kódy demonstrační aplikace.
- Testovací obrázky.